RC25049 (W1009-064) September 27, 2010
Computer Science

IBM Research Report

A Survey of Enterprise View Models

Grady Booch
IBM Research Division

Tilak Mitra
IBM Global Business Services

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

A Survey of Enterprise View Models

Grady Booch and Tilak Mitra

In this paper, we survey contemporary frameworks for visualizing the architecture of enterprises that use software-intensive systems to make their business manifest. We examine a number of such view models by presenting their viewpoints - the point of view of different stakeholders holding different concerns – together with their focus, an analysis of their strengths and weaknesses, and their history.

Introduction

It is not uncommon to encounter software-intensive systems composed of tens of millions of lines of code, and while developers may produce code that has a textual representation, there is much, much more to the life of a system than its source code. The code is the truth, but it is not the whole truth.

Similarly, complex sociotechnical organizations – that is to say, enterprises - exhibit a particular complexity in the dance among wetware (humans), software and hardware. Simply put, an enterprise is a named entity that delivers some product or service with some measurable value. An enterprise is not a product or service, a project, a family of interconnected projects, the architecture of those products and services, or the human organization alone, but rather all these things weave together to form an enterprise.

Complex software-intensive systems and complex enterprises share some common characteristics: they are often ultra-large and are composed of many moving parts, some of which may not have a physical manifestation. Therefore, understanding, reasoning about, and transforming both kinds of systems is incredibly difficult.

Modeling Complex Systems

Humans deal with complexity by abstracting, and so we build models that represent these abstractions to help us understand, reason, and transform.

In the space of civil architecture, standard notations and views have emerged as best practices. Similarly, for software-intensive systems, previous decades have given us flow charts and then data flow diagrams, both developed as a means of projecting a visual metaphor to the interesting parts of a software-intensive system. In present times, the Unified Modeling Language (UML) serves a similar purpose, best suited to contemporary forms of software architecture because they have grown up together. For enterprise systems, standard notations and views are still very much evolving, although many approaches have been proposed, approaches have been widely influenced both by business organizational theory as well as software architecture.

Still, we are unsatisfied: while notations such that the UML provides may offer us a basic vocabulary for visualizing, we are still left with the questions of what do we view and how do we view these things at scale.

Enterprise Architecture and Technical Architecture

By the phrase “software-intensive system” we mean that we care about software and hardware and wetware, although the really interesting problems come about when one addresses systems of these systems. Now, in this world, the most important artifact to which one attends to is the raw, running naked code that runs on hardware and that (most often) interacts with the wetware (that is to say, humans). For these purposes, all other artifacts are secondary or tertiary, although this does not mean that they are inconsequential. Rather, rather it is important to accept these other artifacts – models and tests and requirements and so forth – as critical and necessary, for they help one to deliver the right system at the right time to the right stakeholders with the right balance of cost and value.

Software-intensive systems are a primary (but not the only) mechanism used by enterprises to carry out their mission. Software might be at the very soul of the enterprise (such as with eBay or Amazon), or it might be the product or service itself (such as Intuit), or it might be the way it touches the world (such as Bank of America or Hilton).

That being said, from our experience, EA (enterprise architecture) is not TA (technical architecture), and although the two share the noun "architecture" they are different – but related - beasts. Whereas EA attends to the architecture of a business that uses technology, TA attends to the architecture of the software-intensive systems that support that business. That may sound like just word play, but there are significant differences. Each of these domains - that of the business and that of the system - has fundamentally different stakeholders with different perspectives and different viewpoints. The fact that the both share some aspects of terminology and concerns and even notation is good, but can be confusing in the dialog between those two worlds.

Let’s begin by understanding what each of those two worlds has to say about themselves. The Open Group recently conducted a survey of EA definitions, and the one that received the most votes defined enterprise architecture as “the continuous practice of describing the essential elements of a sociotechnical organization, their relationships to each other and to the environment, in order to understand complexity and manage change.” In contrast, Len Bass and his colleagues define a system’s software architecture as “the structure or structures of the system which comprises the software elements, the externally visible properties of those elements, and the relationships among them.” While at first glance it may seem that both definitions are trying to say the same things – both are all about things and connections - there’s a glaring difference: EA focuses on the “sociotechnical organization” while TA focuses on the “software elements.”

Put another way, TA focuses on systems that interact with humans, whereas in the world of EA, humans are a central element in the vocabulary of things that are architected.

View Models

What we view and how we view these things at scale is the concern of view models. As Wikipedia notes, “a view model or viewpoints framework in systems engineering and software engineering is a framework which defines a coherent set of views to be used in systems analysis and/or the construction of an enterprise architecture. A view is a representation of a whole system from the perspective of a related set of concerns. View model provides guidance and rules for structuring, classifying, and organizing architectures.”

One does not need a set of blueprints to build a doghouse; you generally just do it from the materials lying around, or you may use sketches on a scrap of paper to guide you. Move up to the scale of a house and you certainly need blueprints (and building codes will typically require them) presenting the structure from several points of view. A high rise requires many more views and much more detail. Conducting urban planning or attempting an urban transformation is wickedly hard, and while there may be standard ways of abstracting these things, it stresses the human ability to cope.

In the world of TA, IEEE standard 1471 introduces the concept of a view as a means of reasoning about a system’s architecture from the perspective of different stakeholders with different concerns, but it does not specify what views one should use. Kruchten’s 4+1 model view is perhaps the most well known instance of 1741, an effort which Rozanski and Wood’s work on viewpoints and perspectives builds on. In both of these view models, there are ways to visualize and reason about the ‘software elements” that a software architect or code warrior worry about: use cases, class structures, process interactions, component organization, network topology, the patterns that shape them all.

In the world of EA, there are, by our count, well over two dozen frameworks, some of which are view models although most mix representation and process. Of these, perhaps the most well known (but not necessarily the most used) are the Zachman Framework, the Open Group Architecture Framework (TOGAF), and the Department of Defense Architecture Framework (DoDAF). Each of these frameworks are, to put it delicately, large: Zachman has 36 views, DoDAF has four basic view sets with many specific views, and TOGAF offers a large taxonomy of views for a different (but related) set of areas. In addition to these three dominant EA frameworks there are also several company-proprietary ones, government-sponsored ones, and a sprinkling or more consultant-driven frameworks.

The fact that there are so many EA frameworks in flight relative to the much smaller number of TA frameworks is very telling. The simplicity of the TA

framework market suggests some degree of stability or closure in how to view a software-intensive system. On the other hand, while the vibrancy in the EA market suggests a vigorous and active exchange of ideas, history shows that the market will, over time, decide and reduce the number of active and pragmatic EA frameworks to a much smaller number. The reason we are where we are, we believe, is because business practices are still coming to grips with how best to architect an enterprise that contains a large amount of automated gorp that in the past would have been carried out by people. Furthermore, business themselves are in tremendous flux, not just owing to the continuous and rabid injection of technology, but also owing to the angry global markets, both of which make this an incredibly turbulent domain that is in search of an anchor.

Still, to be very clear, both worlds – EA and TA - must co-exist and both must interoperate. SOA actually has some interesting traction here, because on the one hand, architecting a business around the services it provides and architecting a software-intensive system that makes manifest those services are shared goals of the enterprise and the technology.

Speaking of this concept of two worlds, in the dialog between science and religion, there's a related notion of two worlds: science has some things about which it may speak with authority and faith has some things about which it may speak with authority, but when science tries to answer questions of faith (why is the world the way it is) or vice versa (is there or is there not a randomness in the laws of the universe) then conflict arises. Not to diminish the complex texture of the dance between science and religion - if you want to go there, then the Templeton Foundation is one place to start, although Richard Dawkings has some things to say about that too - but our observation is that EA and TA are similarly of two worlds. Most contemporary economically-meaningful enterprises use software-intensive systems to carry out their mission, and so there is and should be this jiggling between the architecture of the business (as it uses technology) and the architecture of the software-intensive system (as it serves and leads the business).

The good news is that there is considerable overlap in the vocabulary of EA and TA. In both we can speak of services and stories and machines and human and pretty much mean the same things. But, EA has some things about which it may speak with authority (such as business processes) and TA can as well (such as the patterns that shape societies of classes) but we run into dangerous territory when EA tries to speak of things that the software architect cares about or when TA tries to speak of things a human resource agent cares about.

In our experience it's a mistake to try and extend EA frameworks and notations and processes to attend to the architecture of the software-intensive systems it uses, just as it is a mistake to try and extend TA frameworks and notations and processes to attend to the architecture of the business. There might be some overlap in view and basic modeling elements and processes at a high enough

level of abstraction, but when you get to the details, it becomes too much, and you lose the perspective of what each is trying to do. EA is not a dessert topping and a floor wax and neither is TA.

A Gallery of View Models
In our research survey, we have encountered a multitude of view models.

There are two mainstream general purpose TA frameworks:

- Kruchten 4+1 model view
- Rozanski and Woods Viewpoints and Perspectives

There are three mainstream general purpose EA frameworks:

- Zachman Framework
- The Open Group Architecture Framework (TOGAF)
- The Department of Defense Architecture Framework (DoDAF) and its variation for the UK Ministry of Defense (MoDAF)

There are several company-proprietary EA frameworks:

- Capgemini's Integrated Architecture Framework (IAF)
- IBM's System Description Standard (SDS)
- SAP Enterprise Architecture Framework (EAF)
- Oracle Enterprise Architecture Framework (OEAF)

There are a few government-specific EA frameworks:

- Federal Enterprise Architecture (FEAF)
- European Interoperability Framework for eGovernment (EIF)
- Standards and Architecture for eGovernment Applications (SAGA)
- Commission Enterprise Architecture Framework (CEAF)

Other defense-specific EA frameworks include:

- NATO Architecture Framework (NAF)
- Aterlier de Gestion de l'architecture des Systems d'Information et de Communication (AGATE)

Finally, there are the EA frameworks that have sprung up from smaller consulting practices:

- Orr's Business Enterprise Architecture Modeling Framework (BEAM)
- Extended Enterprise Architecture Framework (E2AF)
- Pragmatic Enterprise Architecture (PeaF)

- Purdue Enterprise Reference Architecture (PERA)
- Gartner Enterprise Architecture Framework
- OBASHI

A number of older EA frameworks have come and gone (C4ISR was the predecessor of DoDAF, for example). Some are variations on a common theme (the Treasury Enterprise Architecture, the National Intelligent Transportation Systems Architecture, the National Weather Service Enterprise Architecture, and the Federal Deposit Insurance Corporation Enterprise Architecture Framework are all variations of the Federal Enterprise Architecture). Some are simply obsolete, the world having moved on (the Cross Government Enterprise Architecture, for example).

The following sections offer a summary and an analysis of the view models we've capriciously and arbitrarily deemed to be the most important and/or the most interesting. Specifically – in alphabetical order – we offer a summary and then an analysis of

- BEAM (Orr's Business Enterprise Architecture Modeling Framework)
- DoDAF (the Department of Defense Architecture Framework)
- FEAF (Federan Enterprise Architecture)
- SDS (IBM's System Description Standard)
- TOGAF (the Open Group Architecture Framework)
- Zachman Framework

BEAM (Orr's Business Enterprise Architecture Modeling Framework)

Description

Business Enterprise Architecture Modeling (BEAM) is a systematic approach for developing enterprise architecture for an organization of any size. BEAM is a practical approach to enterprise architecture (EA); getting people in the organization thinking in both the broader as well as longer terms about IT, IT infrastructures, and the core data and systems that IT supports.

The essential underpinning of BEAM is in a business and data/information driven strategy. It strives to achieve a business driven enterprise architecture that is equally applicable to enterprises in both the public and the private sectors. BEAM provides prescriptive guidance to both the new adoptees of enterprise architecture as well as to organizations that have an existing EA program but are striving to extend the impact of their EA initiative.

The fundamental tenet of BEAM is based on a few essential and interrelated components:

1. A basic Enterprise Systems Feedback Model (ESFM) – brings two important ideas to the study of EA namely:

a. All enterprises are in the business to make something or provide some service, and
b. They survive by adding value.

The closed feedback loop introduced ensures that the product and/or services rendered by the enterprise go through a continuous improvement cycle.

2. A new way of considering business to IT alignment – acknowledges and emphasizes on the fact that 'technology planning' impacts the business planning process. It stresses on the fact that the impact of technology drivers, to drive business opportunities, is an essential management interlock that enables *IT managers* to communicate with the *business managers* thereby including the former as a part of the highest level business decision making process.

3. An extension to the Zachman framework – postulates that an urban/transportation planner is a more realistic analogy of an enterprise architect than the classic building architect analogy that Zachman proposed in his framework. This analogy extension allows to acknowledge the over arching responsibilities of the enterprise architect (e.g. futuristic planning of the entire IT infrastructure and roadmap just like the transportation planner plans for 20 or 30 years looking ahead) and not just focus on a single system in the enterprise, which from an analogy standpoint a building architect would do i.e. focus on only the architecture and infrastructure of one building in the entire urban area. The idea here is to define the role of the enterprise architect commensurate with his responsibilities at the enterprise level rather than just at a per-system level.

4. Treating the enterprise architect as a "committee of skills' – acknowledges the fact that there are multiple roles and skills required for the EA group and is not just a case of a single one-person-fits-all. The basic capabilities that the EA group should possess are:
    a. Ability to get user management, IT management, and vendors to reach solutions that work in both the short and the long term.
    b. Ability to conceptualize, envision and built IT infrastructures by considering where people will be operating in the future and then come up with approaches that best provide a communication and compute infrastructure that is both reliable and secure.
    c. Continuous training, certification and skill enhancement to foster technical vitality so as to keep up with the pace of the demands of the marketplace and the technology advancements.

BEAM is developed by Ken Orr and is governed by the Ken Orr Institute (KOI).

Phases
Unlike virtually all of the other EA model views surveyed here, BEAM does not really incorporate the notion of views, but rather incorporates the following five major phases:

1. Strategic intentions
2. Business architecture
3. Data/information/content architecture
4. Application architecture
5. Technology architecture

These phases can be thought of as activity sets that focus on some of the key views or viewpoints in an EA, as perceived in BEAM; the views/viewpoints being around strategy, business context, data/information, application and technology.

While data/information, application and technology architecture have common definitions and have become stable and standard in their definitions, interpretation and usage, the strategic intentions and the business viewpoints in BEAM deserves a mention.

Strategic Intentions – This phase focuses on capturing the business intentions, objectives, drivers and vision that the top management of the enterprise has defined. The strategy viewpoint is typically consolidated through a series of interviews with stakeholders and top management and forms the basis of the business imperatives that needs realization through IT enablement.

Business architecture - The business architecture view is by far the most critical component of the EA as perceived in BEAM. It is the business architecture view that ties the basic and advanced business capabilities and functions to the rest of the enterprise architecture. BEAM breaks the business architecture into three subareas:

1. Business Context – is a visual representation of the major organizational units, business partners and systems that are interconnected through information exchange that collectively represent the organizations inputs, outputs and outcomes. In reality, and based on the size and complexity of the organization, more than one context diagrams are usually required. The collection of all the context diagrams represents the enterprise business context diagram view that all stakeholders can agree upon. Such diagram views allows everyone involved to express their knowledge which are then rationalized into a common understanding. These context diagrams provide a starting point for the business value maps and business processes.

2. Business value maps – is based off of Michael Porter's business value diagram and provides a classification scheme based on which primary

activities, that define the fundamental value-adding processes by which enterprises produce their set of products or services, are distinguished from a set of supporting activities that provide resources and management for primary activities. The business value maps are a set of canonical business processes, at the highest level, based on the "natural breakpoints" in the business activities thereby allowing the organization to sub-divide their work naturally and enable each organizational unit to work independently of one another and communicate through a well defined set of business interfaces. The business value map is an exceedingly important mechanism to understanding an organization's business framework.

3. Business processes – represent the way things are executed or are expected to be executed in an organization. These processes once formalized and documented act as the baseline business steps that are required to be realized through IT enablement. While the business value maps outlines only the canonical business processes this activity focuses on modeling and documenting the detailed set of activities, tasks and steps that comprise the higher level process.

References

Most of the material in the sections above is obtained directly from

Ken Orr, Bill Roth and Ben Nelson. Business Enterprise Architecture Modeling http://www.cutter.com/content-and-analysis/resource-centers/enterprise-architecture/sample-our-research/index.html.

Analysis

The advantage of BEAM is that it is a more complete EA framework than many of its counterparts, not to say that it is the only one. The completeness comes from the coverage of both the business and the technical disciplines or areas that typically constitute a sociotechnical enterprise. In an EA the focus should be on the operational aspects of the business and BEAM provides

The three foundational tenets of EA are people, process and technology. While BEAM addresses the process and technology aspects it seems to be lacking in the people dimension of EA. BEAM seems to lack the prescriptive guidance on how to address and mitigate the organizational challenges of an enterprise architecture.

DoDAF (the Department of Defense Architecture Framework)

Description

The Department of Defense Architecture Framework (DoDAF) provides framework for developing and representing architecture descriptions to ensure a common denominator for understanding, comparing, and integrating architectures across organizational, joint, and multinational boundaries for the

defense domain. DoDAF establishes data element definitions, rules, and relationships and a baseline set of products for the consistent development of systems, integrated, or federated architectures. DoDAF provides guidance for describing architectures for war fighting operations, business operations, and processes. Utilizing this framework institutionalizes a common approach for architecture description, development, presentation, and integration.

DoDAF is built around a reference model leveraged to organize the enterprise architecture (EA) and systems architecture into complementary and consistent views. The DoDAF defines a set of products that act as mechanisms for visualizing, understanding, and assimilating the broad scope and complexities of an architecture description through graphic, tabular, or textual means. It is especially suited to large systems with complex integration and interoperability challenges, and is apparently unique in the use of one of its views – the Operational View - that details the external customer's operating domain in which the developing system is expected to operate.

The vision for utilization of DoDAF is to:

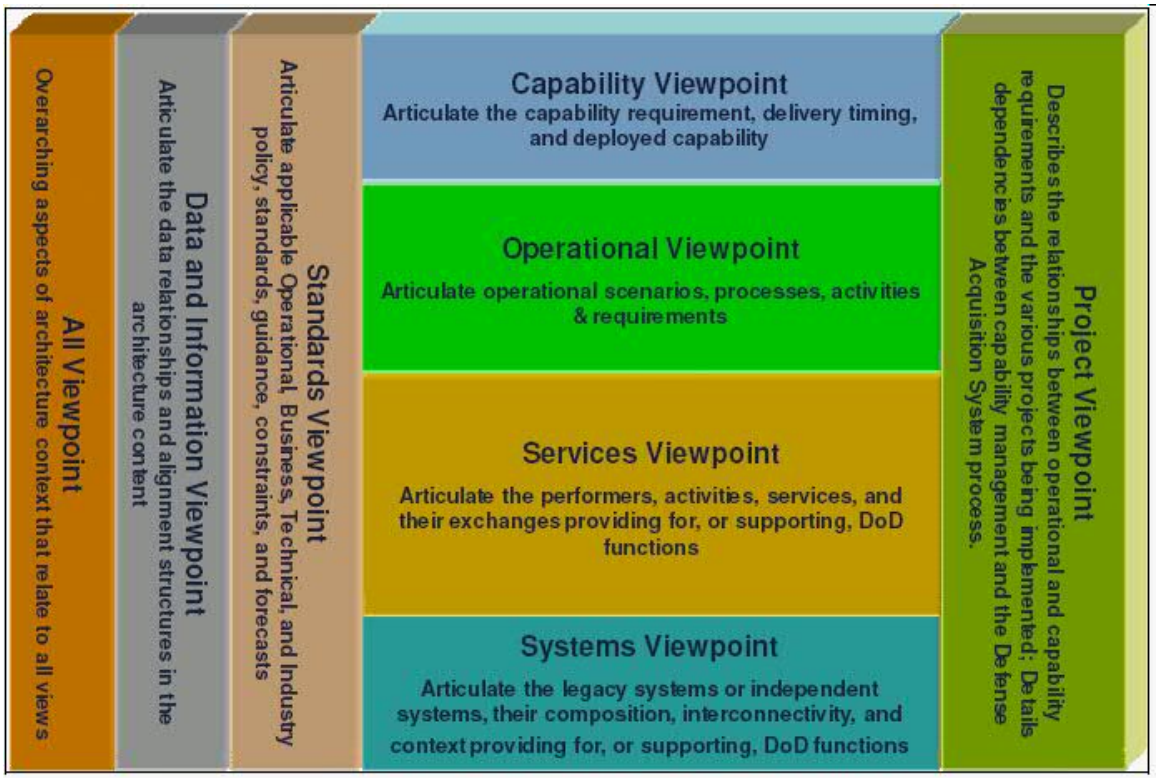
- Provide an overarching set of architecture concepts, guidance, best practices, and methods to enable and facilitate architecture development in support of major decision support processes across all major Departmental programs, Military components, and Capability areas that is consistent and complementary to Federal Enterprise Architecture Guidance, as provided by OMB.
- Support the DoD CIO in defining and institutionalizing the Net-Centric Data Strategy(NCDS) and Net-Centric Services Strategy (NCSS) of the Department, to include the definition, description, development, and execution of services and through introduction of SOA Development.
- Focus on architectural data as information required for making critical decisions rather than emphasizing individual architecture products. Enable architects to provide visualizations of the derived information through combinations of DoDAF-described Models, and Fit-for-Purpose Views commonly used by decision-makers, enabling flexibility to develop those views consistent with the culture and preferences of the organization.
- Provide methods and suggest techniques through which information architects and other developers can create architectures responsive to and supporting Departmental management practices.

As such DoDAF is both an architectural framework and (to a lesser degree) a methodology.

DODAF is governed by the US Department of Defense (DoD).

Views

In DoDAF V2.0, the definition of an Architectural Description is *"a collection of views to document an architecture."*



The All Viewpoint (AV)

The All Viewpoint (AV) models provide information pertinent to the entire Architectural Description, such as the scope and context of the Architectural Description. The scope includes the subject area and time frame for the Architectural Description. The setting in which the Architectural Description exists comprises the interrelated conditions that compose the context for the Architectural Description. These conditions include doctrine; tactics, techniques, and procedures; relevant goals and vision statements; concepts of operations (CONOPS); scenarios; and environmental conditions.

The Capability Viewpoint (CV)

The Capability Viewpoint (CV) captures the enterprise goals associated with the overall vision for executing a specified course of action, or the ability to achieve a desired effect under specific standards and conditions through combinations of means and ways to perform a set of tasks. It provides a strategic context for the capabilities described in an Architectural Description, and an accompanying high-level scope. The models are high level and describe capabilities using terminology, which is easily understood by decision makers and used for communicating a strategic vision regarding capability evolution.

The Data and Information Viewpoint (DIV)

The Data and Information Viewpoint (DIV) captures the business information requirements and structural business process rules for the Architectural Description. It describes the information that is associated with the information exchanges in the Architectural Description, such as attributes, characteristics, and interrelationships.

The Operational Viewpoint (OV)

The Operational Viewpoint (OV) captures the organizations, tasks, or activities performed, and information that must be exchanged between them to accomplish DoD missions. It conveys the types of information exchanged, the frequency of exchange, which tasks and activities are supported by the information exchanges, and the nature of information exchanges.

The Project Viewpoint (PV)

The Project Viewpoint (PV) captures how programs are grouped in organizational terms as a coherent portfolio of acquisition programs. It provides a way of describing the organizational relationships between multiple acquisition programs, each of which are responsible for delivering individual systems or capabilities.

The Services Viewpoint (SvcV)

The Services Viewpoint (SvcV) captures system, service, and interconnection functionality providing for, or supporting, operational activities. DoD processes include warfighting, business, intelligence, and infrastructure functions. The SvcV functions and service resources and components may be linked to the architectural data in the OV. These system functions and service resources support the operational activities and facilitate the exchange of information.

The Standards Viewpoint (StdV)

The Standards Viewpoint (StdV) is the minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements. Its purpose is to ensure that a system satisfies a specified set of operational requirements. The StdV provides the technical systems implementation guidelines upon which engineering specifications are based, common building blocks established, and product lines developed. It includes a collection of the technical standards, implementation conventions, standards options, rules, and criteria that can be organized into profile(s) that govern systems and system or service elements in a given Architectural Description.

The Systems Viewpoint (SV)

The Systems Viewpoint (SV) captures the information on supporting automated systems, interconnectivity, and other systems functionality in support of operating activities**.**

It may be noted that this viewpoint may be phased out in subsequent versions of DODAF with service oriented computing and cloud computing becoming more mainstream technologies.

References

Most of the material in the sections above is obtained directly from

> *The DoD Architecture Repository System (DARS) hosts a publicly downloadable version of the DODAF 2.0 specifications - https://dars1.army.mil.*

Analysis

DODAF is prescriptive in that it provides specific instructions and guidance on its usage; it is informational in that it dedicates an entire volume to the description of architecture products and their usage in the development of an architecture description. The core views - i.e. OV, StdV and SV - are well integrated to form a firm basis to capture the architecture description of any enterprise that are primarily military based.

DODAF is less focused on IT and more so for systems of software-intensive weapons systems. As such, it too lacks the organizational, cultural and social aspects of an organization and its management of change.

Although UML is the modeling language for representing the meta model for DODAF there are differences of opinion in the architecture as well as in the DODAF community as to the coverage and suitability of UML to represent all the DODAF views. This has been the primary reason why DODAF architecture views, through its evolution, has used UML more for the systems view and much less for the business/organizational views.

Another weakness of DODAF results from its principle of inelasticity, that is to say it is unalterable; it follows a direct path with little variance. Within its product descriptions, the framework leaves little room for tolerances or diversions from its intended results.

FEAF (Federal Enterprise Architecture)

Description

The Federal Enterprise Architecture Framework (FEAF) is an enterprise architecture framework for the federal government of the United States that provides a common methodology for the development, usage, and maintenance of IT, for the government.

The purpose of the FEAF is to establish an agency-wide roadmap to achieve an agency's mission through optimal performance of its core business processes within an efficient IT environment. The goal, stated in a different perspective, is to substantially reduce the inconsistency of architectural descriptions across the federal government and to allow a more efficient analysis of duplication and

redundancies of business processes and systems both within and across agencies.

The Federal Enterprise Architecture Framework is an organizing mechanism for managing the development and maintenance of architecture descriptions. The FEAF also provides a structure for organizing federal resources and describing and managing Federal Enterprise Architecture activities. The framework does this by organizing information about the enterprise into various levels, or frames of reference. The top level, Level I, is the highest-level view of the enterprise. The bottom level, Level IV, contains the most detailed information about the enterprise. It partitions the Enterprise Architectures into business, data, application, and technology architectures. The FEAF also takes into account elements of the Zachman Framework and uses the Spewak EA planning methodology.

The goal of the FEAF is to enable the federal government to:

- Organize federal information on a federal-wide scale
- Promote information sharing among federal organizations
- Help federal organizations develop their architectures
- Help federal organizations quickly develop their IT investment processes
- Serve customer needs better, faster, and more cost effectively

The Federal Enterprise Architecture Framework recognizes that architecture development and maintenance requires a process that continually evaluates current conditions and potential solutions. Key aspects of the process include:

- Obtaining executive buy-in and support,
- Establishing a management structure that outlines various roles and activities to facilitate the development of the EA,
- Defining an architecture process and approach,
- Developing both baseline and target EAs,
- Developing a gap analysis to create a sequencing plan to transition systems, applications, and business processes,
- Using the enterprise architecture to prioritize implementation decisions and investments in
- organizational change, and
- Managing the change of the Enterprise architecture over time as the agencies needs are continuously changing and evolving.

FEAF is built around five foundational architecture models each addressing a specific discipline of enterprise architecture. The models are:

- Performance Reference Model (PRM)
- Business Reference Model (BRM)
- Service Component Reference Model (SRM)

- Data Reference Model (DRM)
- Technical Reference Model (TRM)
-

The FEAF method of enterprise architecture is built around the following fundamental tenets:

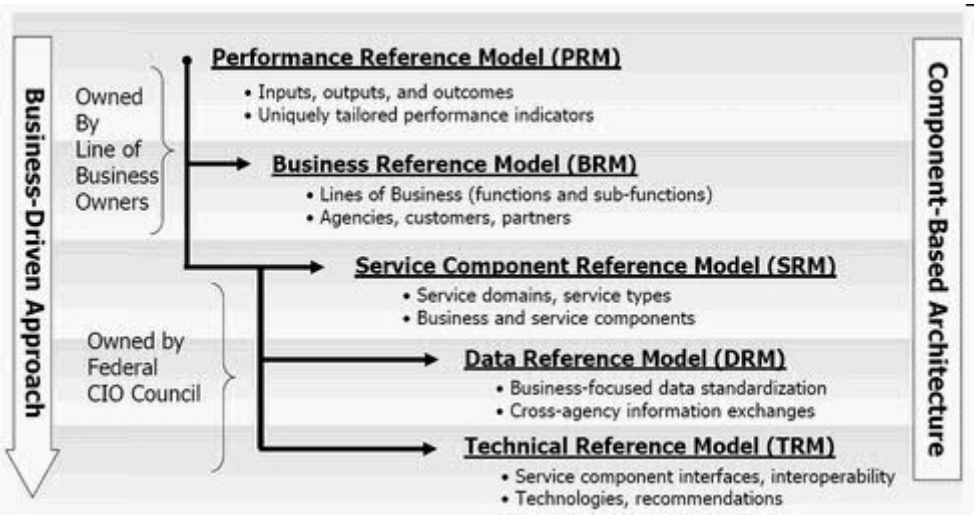
- Architecture Drivers – represents an external stimulus that causes the Federal Enterprise Architecture to change
- Strategic Direction – ensures that changes are consistent with the overall government direction
- Current Architecture – represents the current state of the enterprise or agency. Full characterization may be significantly beyond its worth and maintenance.
- Target Architecture – represents the target state for the enterprise within the context of the strategic direction.
- Transitional Processes – these processes apply the changes from the current architecture to the target architecture in compliance with the architecture standards, such as various decision making or governance procedures, migration planning, budgeting, and configuration management and change control.
- Architectural Segments – these focus on a subset or a smaller enterprise within the total enterprise.
- Architectural Models – provide the documentation and the basis for managing and implementing changes in the enterprise.
- Standards – Include agency adopted standards (both mandatory and voluntary) including best practices and various open standards, all of which focus on promoting interoperability

FEAF has four levels, each of which (from lower to higher) decomposes the above mentioned components into lower levels of granularity thereby resulting in a detailed exposition of the federal enterprise architecture.

FEAF is governed by the US Office of Management and Budget.

Views

FEAF defines five architecture models that form the basis of the current and target architecture views of a federal agency.

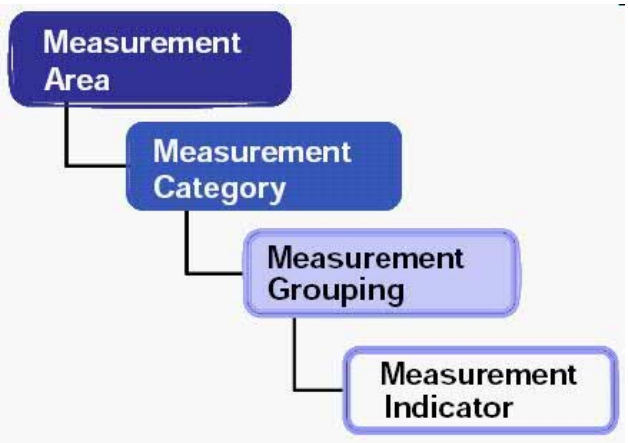


Performance Reference Model (PRM)

The PRM is a standardized framework to measure the performance of major IT investments and their contribution to program performance. The PRM has three main purposes:

1. Help produce enhanced performance information to improve strategic and daily decision-making;
2. Improve the alignment — and better articulate the contribution of — inputs to outputs and outcomes, thereby creating a clear "line of sight" to desired results; and
3. Identify performance improvement opportunities that span traditional organizational structures and boundaries

The PRM is structured around Measurement Areas, Measurement Categories, Measurement Groupings, and Measurement Indicators.



- Measurement Areas – the high-level organizing framework of the PRM capturing aspects of performance at the output levels. This layer is directly linked to the performance objectives established at the agency and

program levels. The PRM includes six measurement areas: Mission and Business Results, Customer Results, Processes and Activities, Human Capital, Technology, and Other Fixed Assets.

- Measurement Categories – collections within each measurement area describing the attribute or characteristic to be measured. For example, the Mission and Business Results Measurement Area include three Measurement Categories: Services for Citizens, Support Delivery of Services, and Management of Government Resources, corresponding to the Lines of Business in the BRM.
- Measurement Groupings – further refinement of categories into specific types of measurement indicators. For the Mission and Business Results Measurement Area, these groupings align to the Sub-functions of the BRM.
- Measurement Indicators – the specific measures, e.g., number and/or percentage of customers satisfied, tailored for a specific BRM Line of Business or Sub-function, agency, program, or IT initiative.

Business Reference Model (BRM)

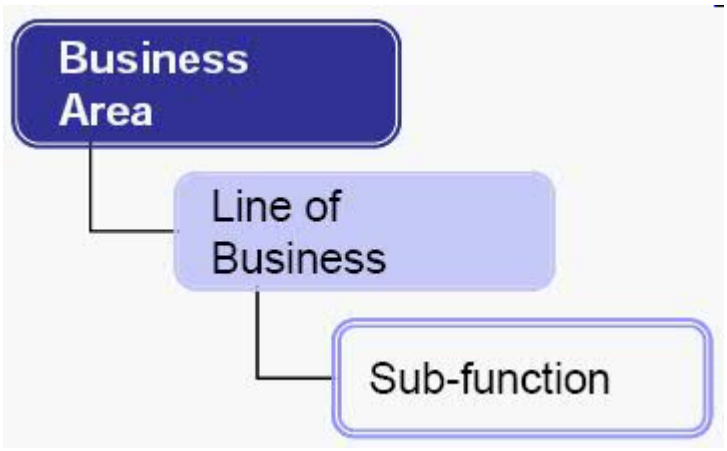
The BRM a function-driven framework for describing the business operations of the Federal Government and is independent of the agencies which perform them. This business reference model provides an organized, hierarchical construct for describing the day-to-day business operations of the Federal government using a functionally driven approach. The BRM is the first layer of the Federal Enterprise Architecture and it is the main viewpoint for the analysis of data, service components and technology.

The BRM is broken down into four areas:

- Services For Citizens
- Mode of Delivery
- Support Delivery of Services
- Management of Government Resources

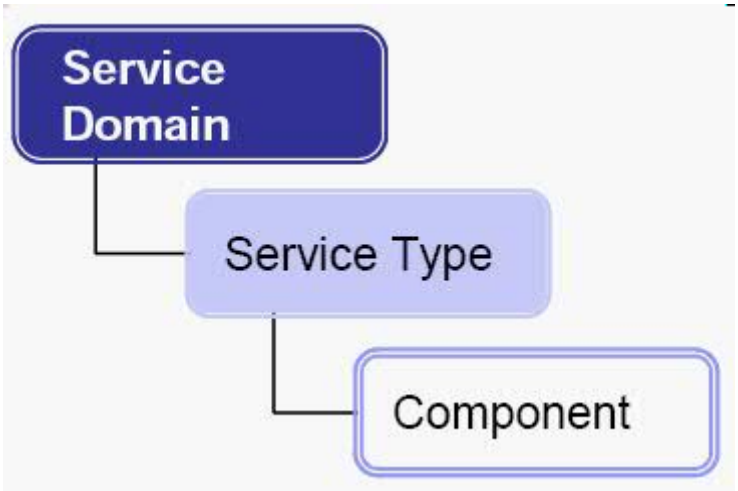
The Business Reference Model provides a framework that facilitates a functional (as opposed to organizational) view of the federal government's LoBs, including its internal operations and its services for the citizens, independent of the agencies, bureaus and offices that perform them. By describing the federal government around common business areas instead of by a stovepiped, agency-by-agency view, the BRM promotes agency collaboration and serves as the underlying foundation for the FEA and E-Gov strategies.

The BRM is structured around Business Area, Line of Business, and Sub-function.



Service Component Reference Model (SRM)
The Service Component Reference Model (SRM) is a business and performance-driven, functional framework that classifies Service Components with respect to how they support business and/or performance objectives. The SRM is intended for use to support the discovery of government-wide business and application Service Components in IT investments and assets. The SRM is structured across horizontal and vertical service domains that, independent of the business functions, can provide a leverage-able foundation to support the reuse of applications, application capabilities, components, and business services.

Each Service Domain is decomposed into Service Types and each Service Type is decomposed further into components.



Data Reference Model (DRM)
The Data Reference Model (DRM) describes the data and information that support government program and LoBs. This model enables agencies to describe the types of interaction and exchanges that occur between the Federal Government and citizens. The DRM categorizes government information into

greater levels of detail. It also establishes a classification for federal data and identifies duplicative data resources.

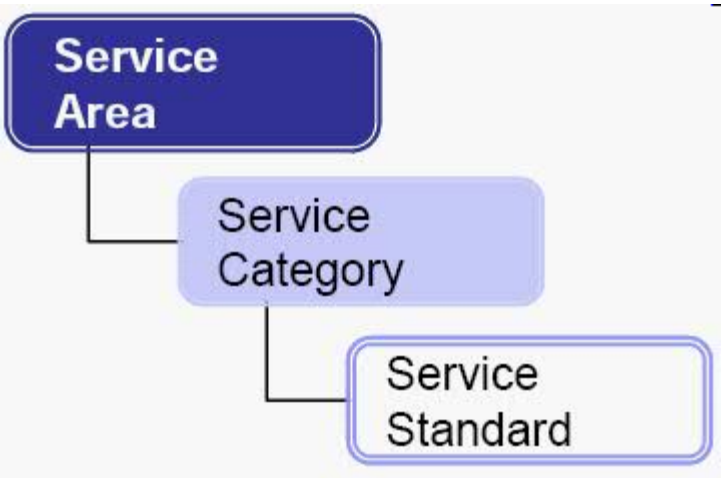
The goal of DRM is to arrive at a common data model that will streamline information exchange processes within the Federal government and between government and external stakeholders.

The DRM is expected to be used as a starting point for data architecture definition along with data modeling and standards that define, refine and constrain the architecture artifacts.

Technical Reference Model (TRM)
The TRM is a component-driven, technical framework categorizing the standards and technologies to support and enable the delivery of Service Components and capabilities.

The TRM consists of:

- Service Areas - represent a technical tier supporting the secure construction, exchange, and delivery of Service Components. Each Service Area aggregates the standards and technologies into lower-level functional areas. Each Service Area consists of multiple Service Categories and Service Standards. This hierarchy provides the framework to group standards and technologies that directly support the Service Area.
- Service Categories - classify lower levels of technologies and standards with respect to the business or technology function they serve. In turn, each Service Category comprises one or more Service Standards.
- Service Standards - define the standards and technologies that support a Service Category. To support agency mapping into the TRM, many of the Service Standards provide illustrative specifications or technologies as examples



References
Most of the material in the sections above is obtained directly from

"A Practical Guide to Developing Federal Enterprise Architecture" - http://www.gao.gov/special.pubs/eaguide.pdf

Analysis
FEAF is a complete and true EA framework. On the organizational aspect, FEAF not only provides guidance on defining a business reference model but also provides prescriptive guidelines on how to define and capture the business performance measures which form the basis of benchmarking the enterprises' success criteria. Being complete renders it to be used more pervasively and hence it enjoys a wide tooling support some examples of which are BEA Aqualogic, IBM Rational Systems Architect, Casewise, Adaptive, Troux Metaverse, and so on.

SDS (IBM's System Description Standard)

Description
The System Description Standard (SDS) Semantic Specification aims at providing a clear and unambiguous definition of the concepts involved in modeling the requirements, architecture and validation of systems. The model views and their interrelationships are represented using the UML2 standard specifications.

SDS is based on the principle that System Models can be built at a level of semantic abstraction (e.g. 'component' is a general concept; its usage in the context of a business system makes it a business component whereas in the context of an IT system makes it represent an IT component) that extends its applicability to all aspects or disciplines of the domain of enterprise architecture modeling. It exploits the concept of "qualifiers" in UML2 to make the intended context explicit.

The SDS Semantic Specification organizes the concepts and relationships from the perspective of 'System Solution Models', however the same concepts are leveraged to support the 'Architecture Building Block' models in an IT system, with refinements and additional constraints applied to make the model context specific. SDS relies on its fundamental underpinning principle to limit the usage of the number of fundamental model constructs such that they are applicable in multiple different contexts, to model different types and categories of systems.

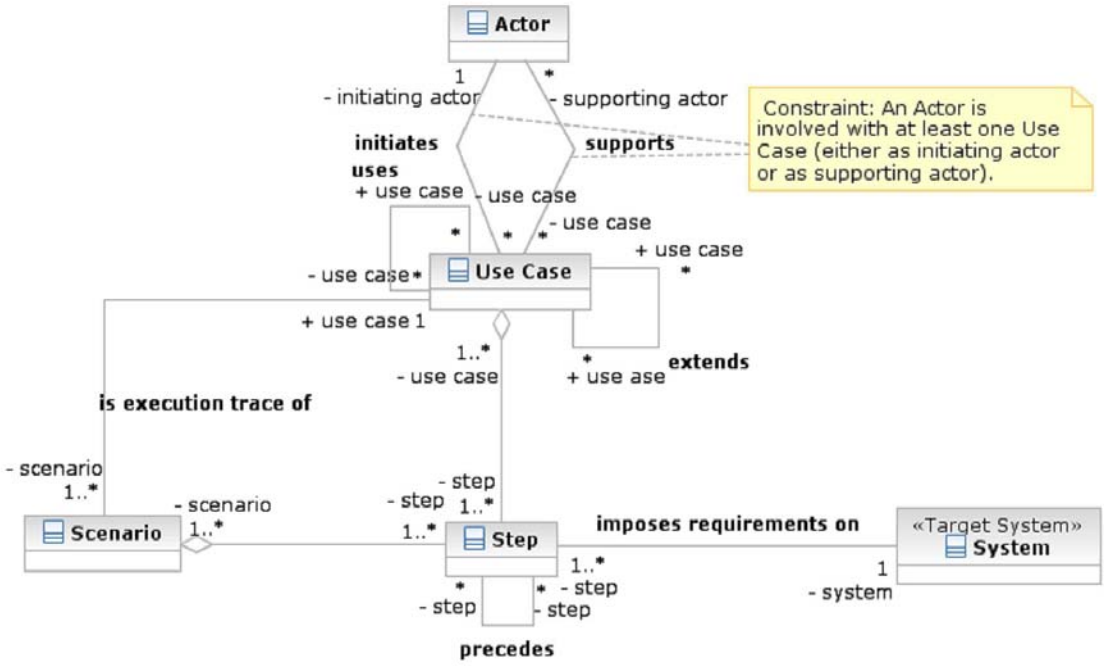
Views
The Architecture Description Standard for systems (SDS) defines a set of four viewpoints namely:

1. Systems Requirement Viewpoint
2. System Functional Viewpoint

3. System Operational Viewpoint
4. System Validation Viewpoint

System Requirement Viewpoint

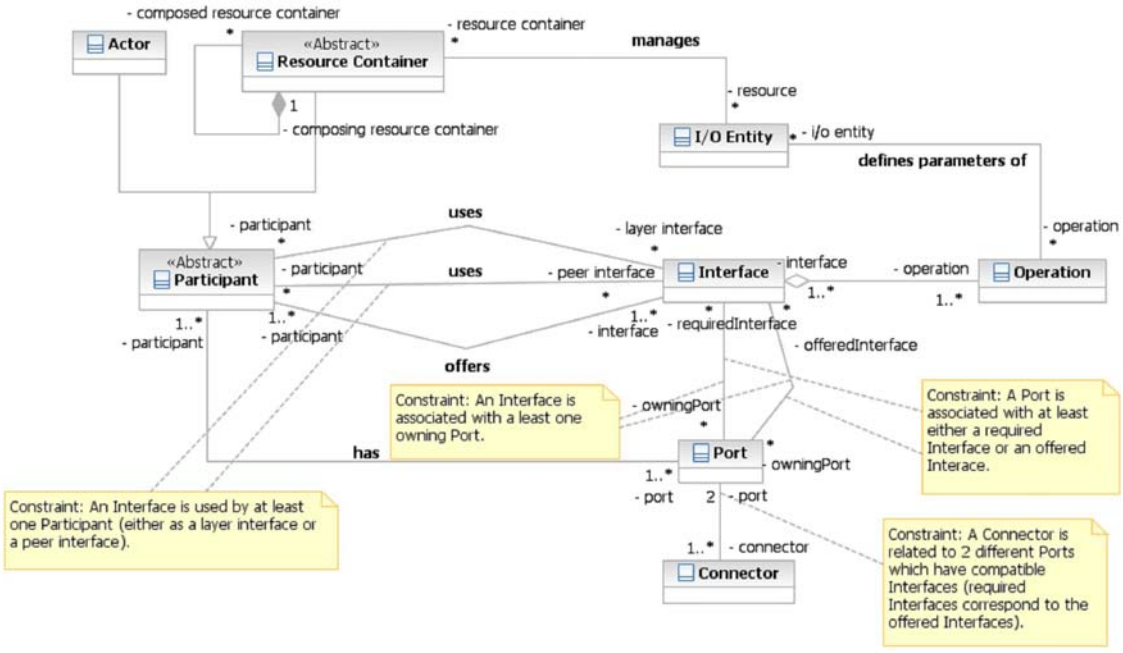
The System Requirement Viewpoint defines a set of elements that may occur in a valid system model as; showing the relationships between them. These core elements form the basis of defining the requirements that a particular system is expected to deliver so that it can participate in a broader context.



- Actor - describes a role a user or an external system plays with respect to the target system.
- Step - an elementary piece of behaviour of the system, observable from either an actor by the target system, or by the target system from an actor.
- Use case - an identifiable and externally observable behaviour of (a part of) the target system.
- Scenario - is the trace of an execution of a use case under well specified circumstances.

System Functional Viewpoint

The System Functional Viewpoint defines a set of elements that may occur in a valid system model supporting the functional viewpoint of the architecture of the system.



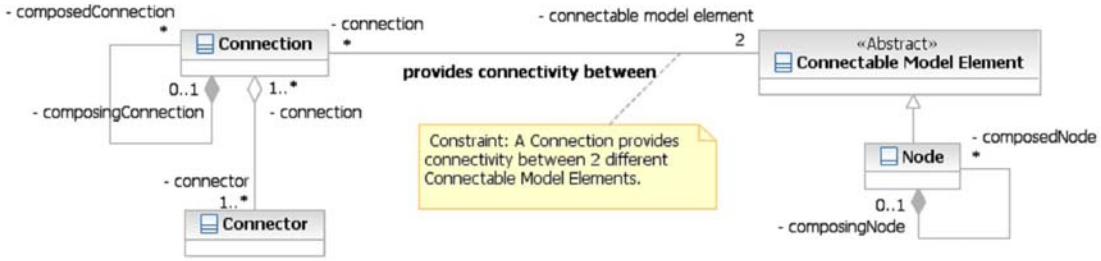
The elements in the figure above represent all the models and their relationships. However, in an actual engineering system, it is not necessary to exploit every model element and all its relationships.

- Participant – is a model element that enables it to interact with other participants via a structured exchange of messages
- Actor - inherit from Participant and can participate in well-defined communications (via the use of Interfaces, Ports and Messages) with other types of Participant such as Resource Containers and other Actors
- Connector - enables the exchange of messages during interactions between resource containers. The actual messages that can flow across the connector are defined via the required and offered interfaces associated with the ports
- Input/output (I/O) Entity - is anything exchanged either internally, i.e. between parts of a system, or externally, i.e. between parts of the system and its environment
- Interface - specifies a set of operations offered by *Participants*. Interfaces are used and offered by *Participants*. They consist of one or more *Operations* and can be used to define *Ports*. Interfaces can be used to capture the role a participant plays in a collaboration.
- Opertaion - defines the message which can trigger a specific behaviour of a participant. An operation is grouped into one or more *Interfaces*. The parameters of an *Operation* are defined via the associated *I/O Entities*.
- Port - defines the messages, via its associated interfaces and their operations, which a participant can exchange with another participant during an interaction

- Resource Container - a collection of resources which enable it to deliver a certain function

System Operational Viewpoint

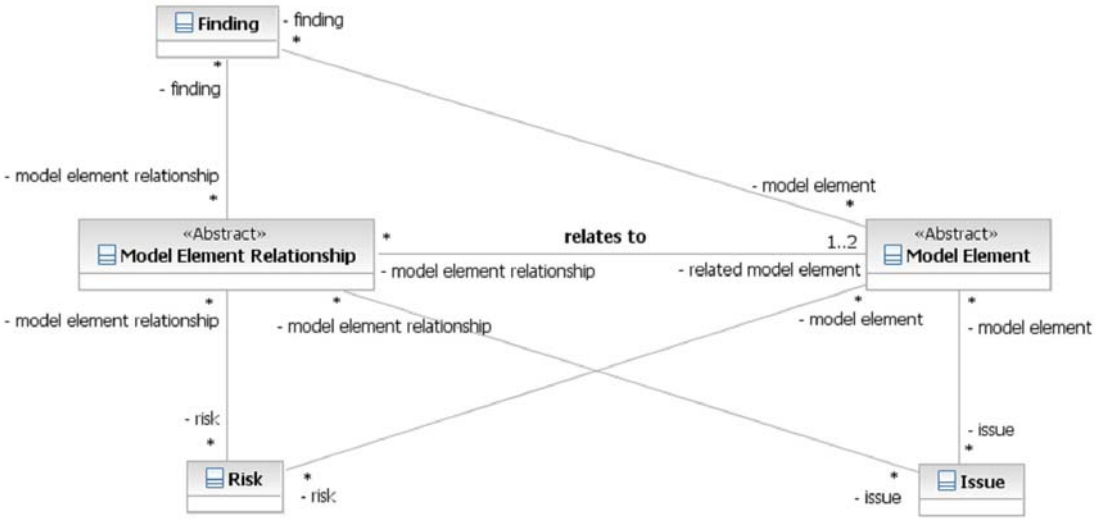
The System Operational Viewpoint defines a set of elements that may occur in a valid system model supporting the operational viewpoint of the architecture of the system. This viewpoint focuses on how the target system is built from its structural elements and integrates into its environment.



- Connection - delivers the required connectivity between connectable model elements. A Connection provides the connectivity between 2 Connectable Model Elements, can compose other Connections, and aggregates Connectors.
- Connector - enables the exchange of messages during interactions between resource containers. The actual messages that can flow across the connector are defined via the required and offered interfaces associated with the ports.
- Node - a collection of resource containers fulfilling a specific responsibility with a certain quality of service within the target system.
- Connectable Model Element - has the capability to be connected to another connectable model element as opposed to a Non-Connectable Model element that does not have the same capability.

System Validation Viewpoint

The System Validation Viewpoint defines a set of conceptual elements that assist in assessing whether a system will deliver its intended functionality with the expected quality of service.



- Finding – defines or captures the result of an investigation.
- Issue – defines a generic term that relates to a matter of concern during the development of a system.
- Risk – defines a potential event of a future situation that may adversely affect the development of the system.
- Model Element - is a part of the system being modeled.
- Model Element Relationship - is a relationship established between one or two model elements.

The Architecture Description Standard for IT Systems (ADS) defines the concepts used in modelling IT systems.
The concepts defined in SDS for modelling systems provide the basis for ADS and are only refined in cases where such redefinition allows for a richer set of relationships and meanings. The result of any redefinition will only be applicable to the IT system's context.
The majority of concepts are equally well applicable to system's as to IT system's contexts.

The Architecture Description Standard for IT Systems (ADS) defines the same 4 viewpoints (as in SDS), namely:

1. IT System Requirement Viewpoint
2. IT System Functional Viewpoint
3. IT System Operational Viewpoint
4. IT Systems Validation Viewpoint

References
Most of the material in the sections above is obtained directly from an IBM internal document

"SDS Semantic Specification R3.0" IBM, 2010.

Analysis
SDS focuses mainly on the technical aspects of the sociotechnical enterprise (with no surprise; it is heavily influenced by Kruchten's 4+1 model view). As such, it does not address the social and organizational challenges and capabilities that a EA framework ought to address.

TOGAF (the Open Group Architecture Framework)

Desciption
TOGAF is an architecture framework – **The Open Group Architecture Framework**. TOGAF is a tool for assisting in the acceptance, production, use, and maintenance of architectures. It is based on an iterative process model supported by best practices and a reusable set of existing architectural assets.

TOGAF is developed and maintained by The Open Group Architecture Forum. The first version of TOGAF, developed in 1995, was based on the US Department of Defense Technical Architecture Framework for Information Management (TAFIM).

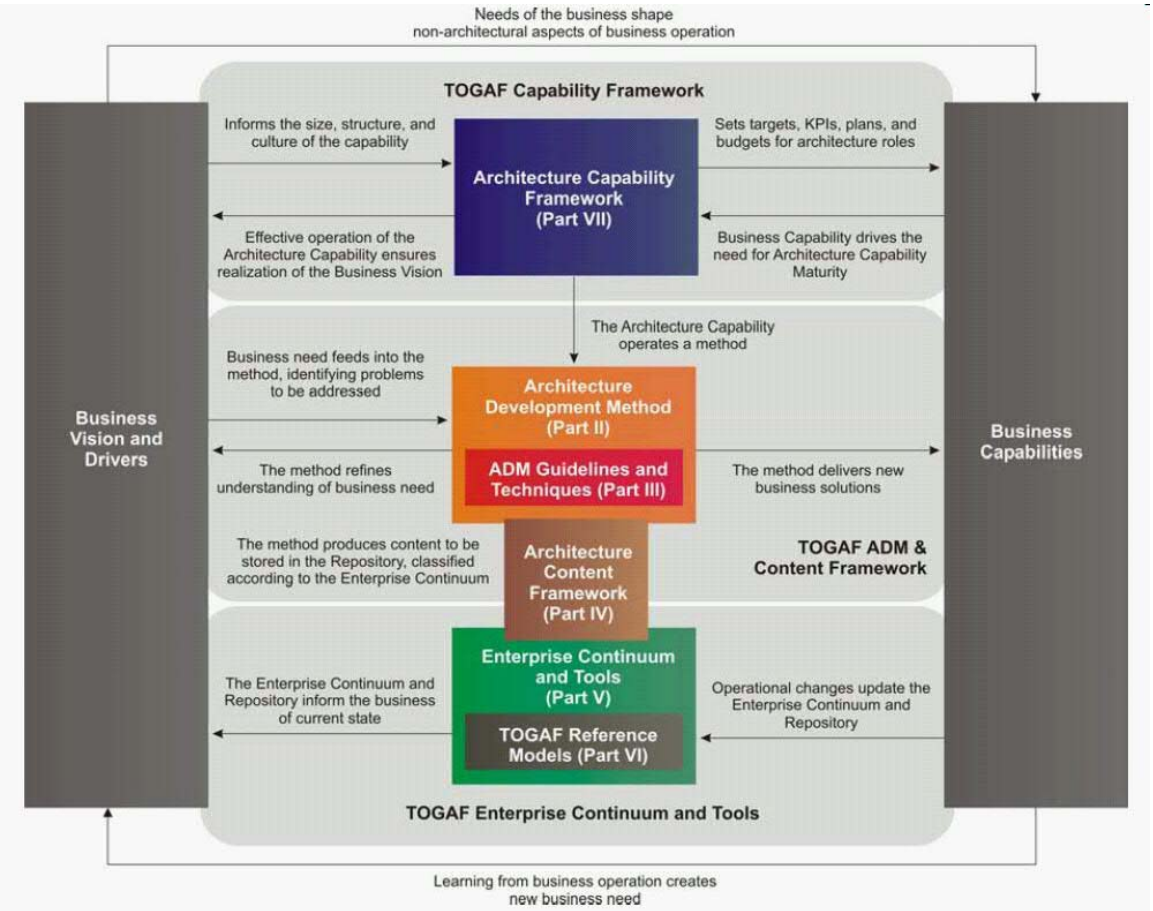
TOGAF 9, the current version, can be used for developing a broad range of different enterprise architectures. TOGAF complements, and can be used in conjunction with, other frameworks that are more focused on specific deliverables for particular vertical sectors such as Government, Telecommunications, Manufacturing, Defense, and Finance. The key to TOGAF is the method – the TOGAF Architecture Development Method (ADM) – for developing an enterprise architecture that addresses business needs.

TOGAF 9 covers the development of four related types of architecture which are the commonly accepted dimensions of an enterprise architecture, namely Business Architecture, Data Architecture, Application Architecture and Technology Architecture.

The Business Architecture focuses on business strategy, governance, organization and key business processes; the Data Architecture addresses the structure of the organization's logical and physical data assets and data management services; the Application Architecture focuses on developing a blueprint for the individual applications to be deployed, their interactions and relationship to the core business processes; the Technology Architecture focuses on the logical software and hardware capabilities that are required to support the deployment of business, data and application services.

TOGAF is governed by The Open Group of which more information is available through its official website http://www.opengroup.org.

TOGAF represents the structure and content of architecture capability within an enterprise. It contains six capabilities that collectively form the meta-model for an enterprise architecture as defined by TOGAF.



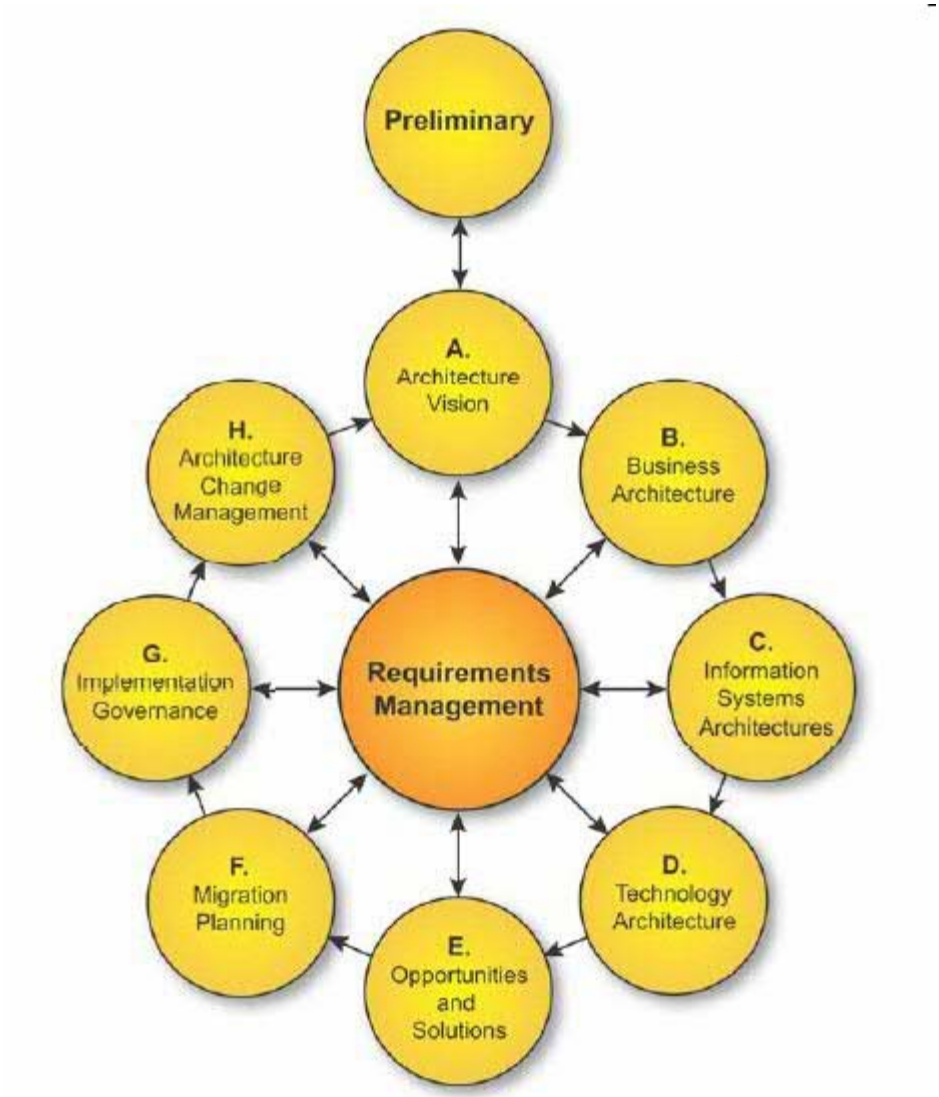
The Architecture Development Method (ADM)

The **ADM** describes how to derive an organization-specific enterprise architecture that addresses business requirements. The ADM is the major component of TOGAF and provides guidance for architects on a number of levels:

- It provides a number of architecture development phases (Business Architecture, Information Systems Architectures, Technology Architecture) in a cycle, as an overall process template for architecture development activity.
- It provides a narrative of each architecture phase, describing the phase in terms of objectives, approach, inputs, steps, and outputs. The inputs and outputs sections provide a definition of the architecture content structure and deliverables (a detailed description of the phase inputs and phase outputs is given in the Architecture Content Framework).

- It provides cross-phase summaries that cover requirements management.

The following pictorial depicts the difference phases of the ADM.



ADM Phase	Activities
Preliminary Phase	• Prepare the organization for successful TOGAF architecture projects • Undertake the preparation and initiation activities required to meet the business activities required to meet the business directives of a new enterprise architecture, includein gthe definition of an organization-

	specific architecture framework and tools, and the definition of principles
Requirements Management	• Every stage of a TOGAF project is based on and validates business requirements • Requirements are identified, stored and fed into and out of the relevant ADM phases, which dispose of, address and prioritize requirements
Phase A: Architecture Vision	• Set the scope, constraints, and expectations for a TOGAF project. • Create the Architecture Vision. Define stakeholders. Validate the business context and create the Statement of Architecture Work. Obtain approvals.
Phase B: Business Architecture Phase C: Information Systems Architecture (Application & Data) Phase D: Technology Architecture	• Develop architectures at three levels: Business, Information Systems, Technology In each case, develop the Baseline and Target Architecture and analyze gaps.
Phase E: Opportunities and Solutions	• Perform initial implementation planning and the identification of delivery vehicles for the building blocks identified in the previous phases. • Identify major implementation projects, and group them into Transition Architectures.
Phase F: Migration Planning	• Analyze cost benefits and risk. • Develop detailed Implementation and Migration Plan.
Phase G: Implementation Governance	• Provide architectural oversight for the implementation. • Prepare and issue Architecture Contracts (Implementation Governance Board). Ensure that the implementation project conforms to the architecture.

Phase H: Architecture Change Management	• Provide continual monitoring and a change management process to ensure that the architecture responds to the needs of the enterprise thereby maximizing the value of architecture to the business

ADM Guidelines and Techniques

ADM Guidelines and Techniques provides a number of guidelines and techniques to support the application of the ADM. The guidelines address adapting the ADM to deal with a number of usage scenarios, including different process styles (e.g., the use of iteration) and also specific specialty architectures (such as security). The techniques support specific tasks within the ADM (such as defining principles, business scenarios, gap analysis, migration planning, risk management, etc).

The Architecture Content Framework

The Architecture Content Framework provides a detailed model of architectural work products, including deliverables, artifacts within deliverables, and the Architecture Building Blocks (ABBs) that deliverables represent.

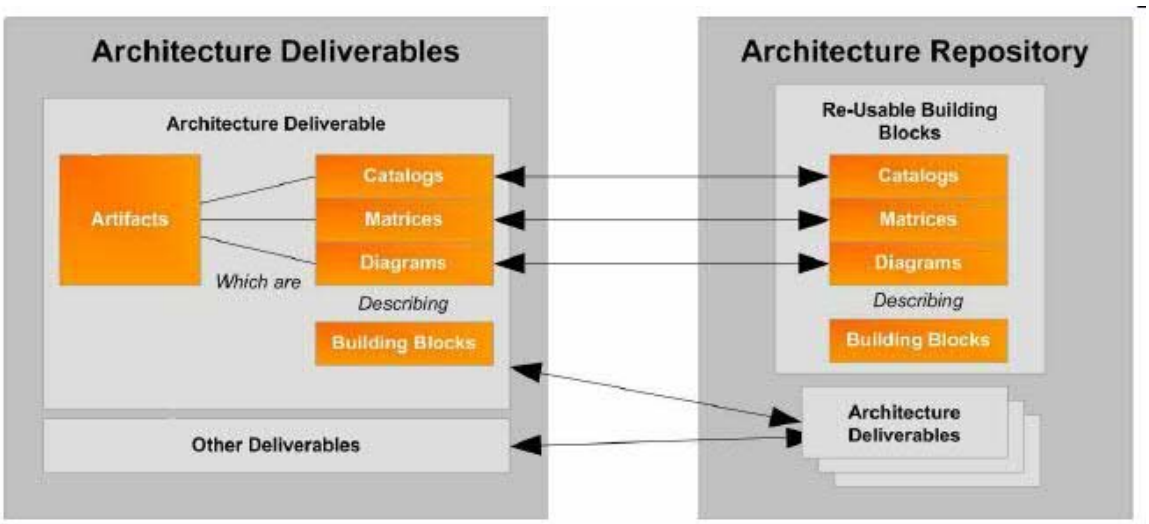
The Architecture Content Framework allows TOGAF to be used as a stand-alone framework for architecture within an enterprise. However, other content frameworks exist (such as ArchiMate and the Zachman Framework) and it is expected that some enterprises may opt to use an external framework in conjunction with the ADM instead. In these cases, the Architecture Content Framework provides a useful reference and starting point for TOGAF content to be mapped to the meta-models of other frameworks.

The Architecture Content Framework uses the following three categories to describe the type of architectural work product within its context of use:

- A deliverable is a formal work product that is contractually specified, and would normally be reviewed, agreed, and signed off by its stakeholders. Deliverables often represent the output of projects.
- An artifact is a more granular architectural work product that describes architecture from a specific viewpoint. This would include such things as a use-case specification, a list of architectural requirements, or a network diagram. Artifacts are generally classified as either catalogs (lists of things), matrices (showing relationships between things), or diagrams (pictures of things). An architectural deliverable may contain many artifacts.

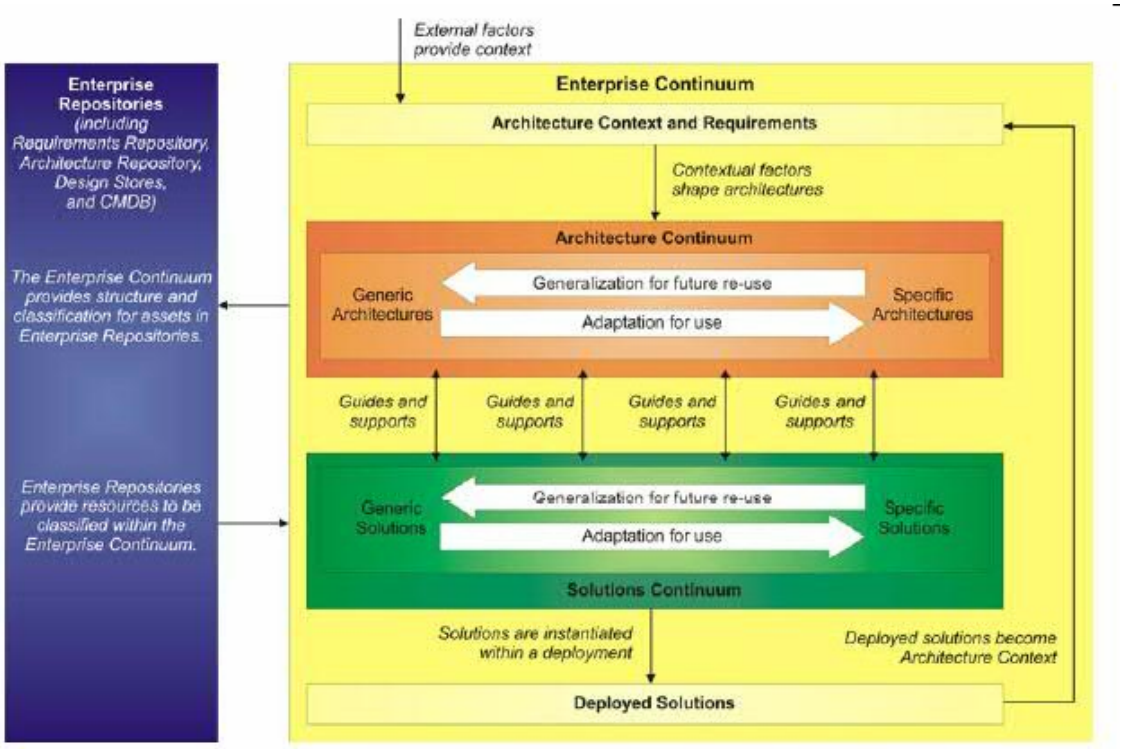
- A building block represents a (potentially re-usable) component of business, IT, or architectural capability that can be combined with other building blocks to deliver architectures and solutions.

The figure below illustrates the relationships between deliverables, artifacts and building blocks.



The Enterprise Continuum

The Enterprise Continuum provides a model for structuring a virtual repository and provides methods for classifying architecture and solution artifacts, showing how the different types of artifacts evolve, and how they can be leveraged and re-used. This is based on architectures and solutions (models, patterns, architecture descriptions, etc.) that exist within the enterprise and in the industry at large, and which the enterprise has collected for use in the development of its architectures.



The Enterprise Continuum supports two general ideas: re-use where possible, especially the avoidance of re-invention, and an aid to communication. The assets in both the Architecture and Solutions Continuums are structured from generic to specific in order to provide a consistent language to effectively communicate the differences between architectures. Understanding where you are in the continuum helps everyone to communicate effectively and eliminate ambiguity.

TOGAF Reference Models

TOGAF provides two reference models for possible inclusion in an enterprise's own Enterprise Continuum, namely the TOGAF **Technical Reference Model** (TRM) and the **Integrated Information Infrastructure Model** (III-RM).

The Architecture Capability Framework

The Architecture Capability Framework is a set of resources, guidelines, templates, background information, etc. provided to help the architect establish an architecture practice within an organization.

Views

TOGAF 9 defines a set of sample viewpoints that may be leveraged during the execution of each phase of the ADM. The set of sample viewpoints by phase are listed in the table below:

ADM PHASE	Viewpoints
Preliminary Phase	• Principles Catalog – used to capture principles of the business and architecture principles that describe what a “good” solution or architecture should look like
Architecture Vision	• Stakeholder Map matrix – identifies the various stakeholders for the architecture engagement, their influence over the engagement, and their key questions, issues or concerns that must be addressed by the architecture framework • Value Chain diagram – provides a high level orientation view of an enterprise and how it interacts with the outside world • Solution Concept diagram – illustrates concisely the major components of the solution and how the solution will result in a benefit for the enterprise
Business Architecture	• Organization/Actor catalog – captures a definite listing of all participants that interact with IT, including users and owners of the system • Driver/Goal/Objective catalog – provides a cross organizational reference of how an organization meets its business drivers in practical terms through goals, objectives and (optionally) measures • Role catalog – provides a listing of all authorization levels or zones within the enterprise • Business Service/Function catalog – provides a functional decomposition view in a form that can be filtered and queried upon • Location catalog – provides a list of locations where an enterprise carries out business operations or houses architecturally relevant assets e.g. data centers • Process/Event/Control/Product catalog – functions as a supplement to existing Process Flow Diagrams and provides a hierarchy of all the processes, events that trigger processes, outputs from processes and controls applied to the execution of processes • Contract/Measure catalog – provides a listing of all agreed service contracts and (optionally) the measures attached to those contracts • Business Interaction matrix – depicts the relationships and interactions between the organizations and the business functions within

	an enterprise • Actor/Role matrix – illustrates which actor performs which roles alongwith supporting security and skill requirements • Business Footprint diagram – describes the links between business goals, organizational units, business functions and services and maps these functions to the technical components that deliver the required capability • Business Service/Information diagram – shows information needed to support one or more business service • Functional Decomposition diagram – provides a single pictorial representation of the capabilities of an organization that are relevant to the consideration of an architecture • Product Lifecycle diagram – assists in understanding the lifecycle of key business entities • Goal/Objective/Service diagram – defines the way a business service contributes to the achievement of a business vision or strategy • Business Use-Case diagram – display the relationship between the consumers and providers of business services • Organization Decomposition diagram – describes the relationships between actors, roles and locations within an organization treel • Process Flow diagram – depicts a sequential flow of activities and tasks that make up the enterprise business processes • Event diagram – depicts the relationship between events and processes
Data Architecture	• Data Entity/Data Component catalog – provides a list of all the data entities that are used across the enterprise including data components where the data entities are stored • Data Entity/Business Function matrix – depicts the relationship between data entities and business functions within an enterprise • System/Data matrix – depicts the relationship between IT systems and the data entities that are accessed and updated by them • Class diagram – depicts the relationships between the key data entities within an enterprise • Data Dissemination diagram – shows the

	relationship between data entities, business services and application components • Data Security diagram – depicts which actor has access to which enterprise data • Class Hierarchy diagram – provides a view to the stakeholders of who is using the data, how, why and when • Data Migration diagram – depicts flow of data from the source to the target applications • Data Lifecycle diagram – assists in managing the enterprise data through all its possible lifecycle stages that spreads from its conception to its disposal within the constraints of the business processes that it serves
Application Architecture	• Application Portfolio catalog – maintain a list of all the applications in the enterprise • Interface catalog – scopes and documents the interfaces between applications. This enables to define, early on, the dependencies between the applications • System/Organization matrix – depicts the relationship between IT systems and organizational units within an enterprise • Role/System matrix – depicts the relationship between IT systems and the business roles that use them within the enterprise • System/Function matrix – depicts the relationship between IT systems and business functions within an enterprise • Application Interaction matrix – depicts the communication relationships between IT systems • Application Communication diagram – depicts all models and mappings related to communication between applications in the metamodel entity. Communications here are typically logical in nature in the sense that they depict intermediary technology only when it is architecturally relevant • Application and User Location diagram – depicts the geographical spread of applications within an enterprise • System Use-Case diagram – displays the relationship between consumers and providers of application services • Enterprise Manageability diagram – shows how one or more applications interact with application and technology components that support

	operational management of a solution • Process/System Realization diagram – used to clearly depict the sequence of events when multiple applications are involved in executing a business process • Software Engineering diagram – provides a development perspective wherein applications are decomposed into their constituent packages, modules, services and operations • Application Migration diagram – helps in identifying application migration from baseline to target application components • Software Distribution diagram – depicts how application software is structured, deployed and distributed across the underlying physical technology infrastructure
Technology Architecture	• Technology Standards catalog – documents the agreed upon enterprise technology standards covering their versions, refresh cycles, lifecycles and so on • Technology Portfolio catalog – lists all the technologies currently in use across the enterprise including hardware, infrastructure software and application software • System/Technology matrix – documents the mapping of business systems to technology platforms • Environments and Locations diagram – depicts which locations hosts which applications, identifies what technologies and/or applications are used at which locations and also identifies locations from which business users typically access the applications • Platform Decomposition diagram – depicts the technology platform that supports the Information Systems Architecture. It also provides an overview of the enterprise technology platform • Processing diagram – focuses on deployable code/configuration and how they are deployed on the technology platform • Networked Computing/Hardware diagram – provides a view of how the application landscape is distributed in the distributed network computing environment. It also provides a good understanding of the technology architecture. • Communications Engineering diagram – provides

	a means of communication between the assets in the technology architecture
Opportunities and Solutions	• Project Context diagram – shows the scope of a work package to be implemented as a part of a broader transformation roadmap • Benefits diagram – shows opportunities identified in an architecture definition, classified according to their relative size, benefit and complexity
Requirements Management	• Requirements catalog – captures things that the enterprise needs to do to meet its objectives

References
Most of the material in the sections above is obtained directly from

TOGAF Version 9 – The Open Group Architecture Framework (TOGAF) available at http://www.opensource.org/togaf

Analysis
TOGAF 9 is not only a vehicle to develop an EA but also a repository of practical, experience-based information on how to go about the process of enterprise architecture. It provides a generic method with which specific sets of deliverables, methods and architectural assets can be integrated. The fact that TOGAF 9 provides a method, a technique and an execution path to build, maintain and sustain a EA makes it the most comprehensive and by far the most widely accepted EA framework in the industry.
TOGAF 9 includes architecture governance into the mainstream EA process. As governance has become an increasingly visible requirement for organizational management, the adoption of governance into TOGAF 9 aligns the framework with current business best practice and also ensures a level of visibility, guidance, and control that will support all architecture stakeholder requirements and obligations.

Although TOGAF is by far the most widely used and adopted EA in the industry - although the degree to which TOGAF is actually in use may be debated - the challenge still remains to harvest a true EA as a asset or a case study that can demonstrate the correct and complete usage of the framework.

Zachman Framework

Description
The Zachman Framework is a framework for enterprise architecture. Created and designed by John Zachman in 1984, it is one of the more popular and widely accepted frameworks that helps organizations conceptualize, define, create and instantiate an enterprise architecture.

The Zachman Framework provides a classification scheme represented as a 6 X 6 matrix manifest as an intersection between two historic classifications – the

columns representing fundamental communications interrogatives (*what,, how, when, who, where and why*) and the rows representing reification transformations (*defining the incremental transformation of an abstract idea into its final instantiated rendition*).

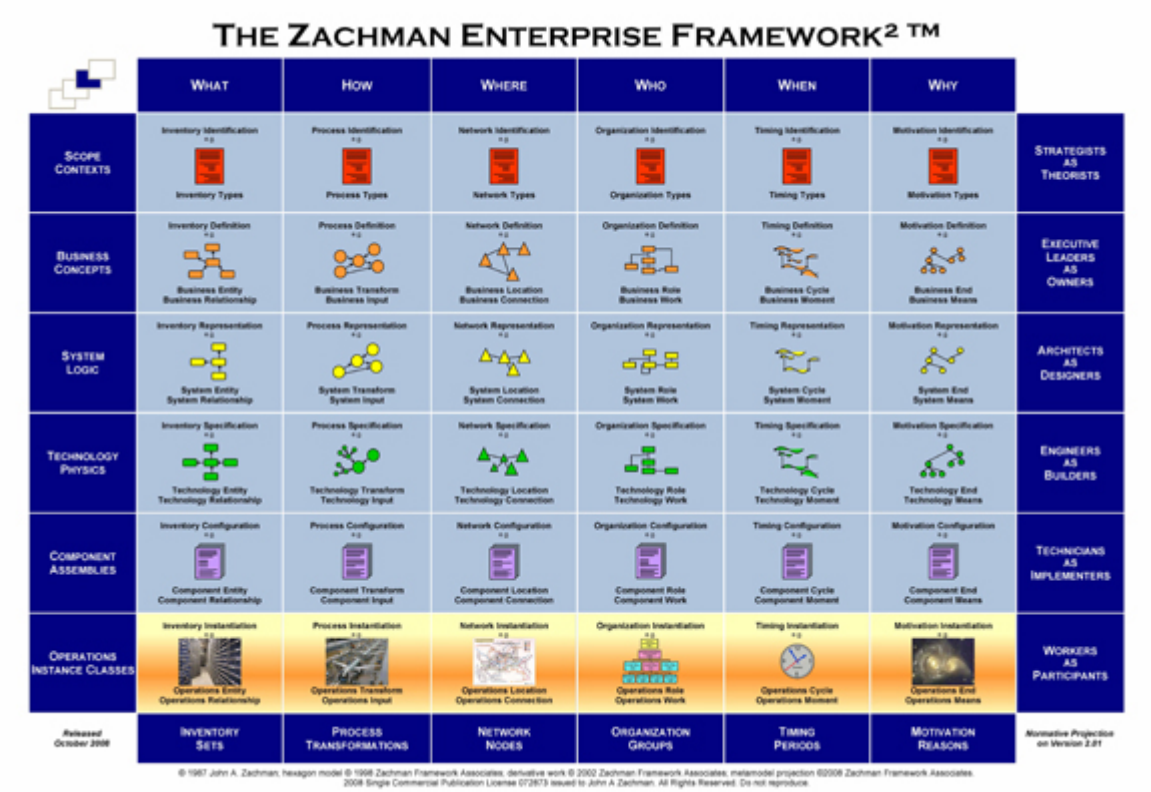
This matrix model provides a progression of movement from row to row with a each offering a more detailed view of how the evolving system moves from earlier phase (e.g. business conceptualization) to the final manifestation of the system through its implementation, deployment, transition and maintenance.

The Zachman Framework is primarily a view model and not a methodology for creating and instantiation of objects; it only represents the ontology for describing the enterprise. This framework embodies an enterprise architecture meta-model which can be instantiated by the application or execution of a process to provide the transformation and make it operational.

The Zachman Framework is governed by The Zachman Institute for Framework Architecture (ZIFA) of which more information is available through its official website http://www.zifa.com.

Views

The views embodied by the Zachman Framework may be represented as a matrix model. This matrix representation of the classification scheme has an enterprise coverage that addresses the business goals, drivers and imperatives and how it is realized through a set of technology manifestations. Figure 1 provides the matrix representation as it is defined in the current version of the framework.



In the matrix, each row addresses a specific view of the solution as it is perceived by a particular stakeholder role who is participant in the systems development process within the enterprise context. This framework postulates that systems are developed by distinct groups with different points of view. Each row, hence, provides a particular perspective of the enterprise architecture and is standalone in such that its coverage, through the communication interrogatives (in the columns) contains sufficient information to complete one aspect or viewpoint of the solution. Although each row is mutually exclusive and self contained it is influenced by constraints and characteristics from other views. More specifically, constraints and assumptions in a view, say V_n , influences the aspects, characteristics and definitions in another view, say V_m , where:

$$n < m$$
$$1 \leq m \leq 5$$
$$2 \leq n \leq 6$$

Thus, the rows and column definitions when applied to an enterprise architecture has the rows identify views of the solution as perceived by various types of stakeholders (business or IT) and the columns identifying the aspects of the architecture.

The top three rows deal with business ideas and are primarily addressed by the business stakeholders, while the bottom three rows deal with operations reality.

This enforces the idea that a transition from a fundamental concept of abstraction has to go through six phases in order to reach reality.

The columns may be mapped as following:

What => Data
How => Function
Where => Network
Who => People
When => Time
Why => Motivation.

Identification View – Scope Context (Row 1)

The focus of this view is on business goals, drivers and requirements.

- What/Data' focuses on the high-level data elements that are related to each function that is under scope.
- 'How/Function' focuses on the high-level business functions in an enterprise
- Where/Network' focuses on the various geographical locations, of the enterprise, where the business functions are carried out.
- 'Who/People' addresses the business stakeholders who own and fund the operations of the business functions.
- 'When/Time' identifies the business events that are related to each business function and the frequency of occurrence of those events.
- 'Why/Motivation' focuses on the business goals, objectives and the key performance measures as it relates to each business function.

Definition View – Business Concepts (Row2)

The focus of this view is primarily on defining the enterprise business architecture in the context of which a system may be developed. As a part of the business architecture definition or instantiation, the business process models for the scope of the system under development are defined. Rationalization of functionality as it pertains to business domains within the business architecture are also addressed which in turn helps in eliminating functional redundancy across the enterprise.

- 'What/Data' - addresses the business data or information that flows through the business process model.
- 'How/Function' - focuses on the business processes that are in scope of the system being considered.
- 'Where/Network' - identifies the enterprise locations in which part (or whole) of the business process is being executed.
- 'Who/People' - identifies the roles of enterprise personnel who either own or are responsible for executing on one or more tasks/steps in the business process.

- ‘When/Time’ - identifies the business events that are related to a business process as a whole and define how and when they are triggered. It also focuses on process improvement activities and their timing in the larger enterprise context.
- ‘Why/Motivation’ - focuses on establishing the policies, procedures, guidelines and standards as they relate to each of the business processes in context.

Representation View – System Logic (Row3)

The focus of this view is for the business stakeholders to define the logical models representing the various aspects of their enterprise. This view also addresses how requirements, for the system under development consideration, should be gathered and formalized.

- ‘What/Data’ - focuses on defining the logical data and models through which data relationships are exhibited.
- ‘How/Function’ - focuses on the logical representation of information systems and their relationships to each other in the enterprise context.
- ‘Where/Network’ - represents the distribution of information systems in the enterprise network.
- ‘Who/People’ - represents the access rights and privileges that are defined for enterprise roles and responsibilities.
- ‘When/Time’ - defines the relationships between the logical/business events and the business actions that they trigger.
- ‘Why/Motivation’ - defines the business rules and policies and guidelines around their variability and applicability to business processes.

Specification View – Technology Physics (Row4)

This is first of the three views that start addressing the business goals, drivers, requirements and capabilities through technology.

- ‘What/Data’ - addresses the persistent technologies and mechanisms that need to be identified and designed for provisioning the data models.
- ‘How/Function’ - focuses on defining the enterprise IT architecture and specific application architectures for each of the systems or applications that are in the scope of development. The application architectures are constrained by the rules and guidelines of the enterprise architecture.
- ‘Where/Network’ - focuses on defining the network devices and their relationships to the physical boundaries of the enterprise locations.
- ‘Who/People’ - specifies the access privileges of users and roles to specific platforms, devices and technologies.
- ‘When/Time’ - provides the definition (specification) of the business events and triggers taking the target technology platform features into consideration.

- ‘Why/Motivation’ - takes the identified business rules (in previous views) and calibrates them using the constraints of the systems through which they are manifested or realized.

Configuration View – Component Assemblies (Row5)

This is a typical designer and implementer’s view that transforms the specification and design details of the system into executable code, making it ready for deployment.

- ‘What/Data’ - addresses data definitions, physical database schema and tables populated with data required for the system to function.
- ‘How/Function’ - addresses executable code that is developed with a target technology platform of choice.
- ‘Where/Network’ - addresses the network deployment model of software and middleware that are configured and distributed on network nodes.
- ‘Who/People’ - configures the access rights and privileges of enterprise roles to run on the target technology platform.
- ‘When/Time’ - addresses the implementation of business events and triggers to the target technology and platform.
- ‘Why/Motivation’ - addresses the implementation of the business rules on the target technology platform.

Instantiation View – Operations Instance Classes (Row6)

This view focuses on the functioning enterprise and what it takes to maintain the technology operations of the running enterprise. It also addresses how evaluation and assessment needs to be performed to analyze and recognize areas of improvement in the operational enterprise.

- ‘What/Data’ - addresses the real time data that is created by the operational systems and how they may be leveraged to analyze the functional enterprise.
- ‘How/Function’ - addresses how to monitor and maintain the applications and systems that collectively constitute the functional enterprise.
- ‘Where/Network’ - addresses how information is being exchanged through the various nodes in the network model and how it may be captured and leveraged for troubleshooting and maintenance of the enterprise systems.
- ‘Who/People’ - focuses on the actual users of the enterprise systems and their conformance to the access rights and privileges.
- ‘When/Time’ - focuses on the real time business events that get propagated through the system and how to provision them for auditing and other analysis.
- ‘Why/Motivation’ - addresses the rationale of monitoring the operational characteristics of functional enterprise along with its technology platforms.

References

Most of the material in the sections above is obtained directly from

"A Framework for Information Systems Architecture." John A. Zachman. IBM Systems Journal, vol. 26, no. 3, 1987. IBM Publication G321-5298. 914-945-3836 or 914-945-2018 fax.

"Extending and Formalizing the Framework for Information Systems Architecture." J.F. Sowa and J. A. Zachman. IBM Systems Journal, vol. 31, no. 3, 1992. IBM Publication G321-5488. 1-800-879-2755.

The Evolution of the Zachman Framework as illustrated in the official ZIFA web stie - http://www.zachmaninternational.com/index.php/ea-articles/100#maincol

Analysis

One of the beauties of the Zachman Framework is that a single row completely represents a specific perspective of the enterprise; full coverage of each of the columns in a row provides a complete model for a single perspective. These models, the ones that are defined for each row, are not independent: rather they are interdependent and interact continuously.

The framework is project or industry agnostic i.e. the six interrogatives are equally applicable to all project or project types. This neutrality enables the framework to be made applicable to enterprises of all shapes and sizes.
The cells in the framework which are the intersection points between perspectives and interrogatives represent a unique artifact which has both context and dimension; the information contained in the cell cannot be found anywhere else throughout the framework.

The framework has its own share of deficiencies and disadvantages. For a complex enterprise the classical method of solving a problem is through the decomposition technique which takes a specific perspective and decomposes the problem domain iteratively so as to arrive at primitive of fundamental solution artifacts. The interrogative approach does not help the cause of decomposition. The framework would have been more widely accepted if it provided more prescriptive guidance in its usage and customization for various project scenarios; it lacks the rules and regulations for its usage. Although it is an EA framework it lacks commensurate techniques around the organizational behavior and challenges that are so much in the forefront of an enterprise architecture.